Arduino Nano as (virtual) keyboard

This paper goes through the development and the setup of a virtual keyboard made using an AVR ATMEGA328P based **Arduino Nano**, that is not compatible with the HID (Human Interface Device) protocol. The Atmel's microcontroller on board, in fact, is connected to the computer through the serial port and not using the Universal Serial Bus (USB) protocol.

AAC Keys

AAC Keys is a lightweight software freely released by the AAC Institute (https://aacinstitute.org/aac-keys/), available both for Windows and Mac OS machines, that is able to read the serial port and *translate* the bits sent from the Arduino Nano to generate events of a virtual keyboard. A software like this one was intended mainly to allow automatic input from external peripherals, such as bar-code readers, but they can work well with every kind of serial data sender; in fact, any type of device with a serial port can be connected through a USB cable/adaptor and work like a keyboard. Actually, AAC Keys also allows to emulate all the keys of the keyboard (not only printable characters and numbers), the combination of several keys and mouse commands.

The following document is focused to the setup of an Arduino Nano (actually, it is the clone with the CH340 driver, but there is no difference) to emulate on Windows (precisely, Windows 10 Pro x64) keyboard events.

From the website of the AAC Institute it is possible to download the software **AAC Keys**; its installation is quite straightforward, as the configuration: the COM port and the baudrate must be set accordingly to the serial communication set for the Arduino. Then, it is all about sending the serial data with the right format, according to the GIDEI, that stands for **General Input Device Emulating Interface**. If the software is set to receive input as ASCII characters, printable characters, namely letters and numbers, can be sent simply as text in a `Serial.print` command. To type specific keys (such as Return, DEL, arrows, etc.) it can be used the so-called **escape sequence**, i.e. the escape key (ASCII 27) followed by the key name and terminated by the period (ASCII 46); between ESC and the desired key it may be necessary (Mac OS) or not (Windows) the comma. Moreover, some

keyboard commands are valid, such as **combine** (type up to 5 keys simultaneously), **hold** (to keep a key until another one is pressed) and the commands **lock** and **rel**, used to hold and release keys. Further details can be obtained at the GIDEI's documentation page (http://park.org/Guests/Trace/pavilion/gideidoc.htm#glance).

As an example, below it is reported the code for Arduino to type the UP arrow when a button is pressed.

```
void setup() {

  pinMode(2, INPUT);
  Serial.begin(9600); // 8N1

}

unsigned long deltat = 50;
unsigned long last = 0;

void loop() {

  while( digitalRead(2) == HIGH ) { // Debouncing

    if (  millis() - last > deltat ) {

      while( digitalRead(2) == HIGH ); // Wait falling edge
      Serial.write(27);
      Serial.print("up."); // Up arrow
    }
  }
  last = millis();

}
```

Export sensor data to Excel

A more interesting application of AAC Keys is to export the data stream sampled by a sensor connected to Arduino in order to log it in a Excel's file, that can be furtherly processed also using Matlab or any other signal processing tool capable of reading such files. The variable containing the ADC reading and its timestamp can be serially sent to the computer, along with arrows combinations to write the data in a two-column file, for example. In the following example, it is reported the code to log the resistance value of

an LDR (Light Dependent Photoresistor). Installing a hot-key software, it would be also possible to send a combination of keys that automatically opens Excel to start the data logging session when the Arduino is plugged in.

```
void setup() {

  Serial.begin(9600);

  // Send Excel's hotkey to open a new file for logging
  // (facoltative)

  // Excel template generation
  Serial.print('Timestamp');
  Serial.write(27);
  Serial.print("right.");
  Serial.println('Sensor value');
  Serial.write(27);
  Serial.print("left.");

  // Ready to log data

}

float Vadc = 0;
float LDR = 0;
unsigned long timestamp = 0;

void loop() {

  // Read the sensor on channel A0 and convert it to voltage
  Vadc = (5*analogRead(A0))/1023;
  timestamp = millis();
  LDR = ( Vadc*10000 ) / ( 5 - Vadc );

  Serial.print(timestamp);
  // Send right arrow key to move to the "Sensor value" column
  Serial.write(27);
  Serial.print("right.");
  Serial.println(LDR); // Write the value and return
  // Send left arrow to move back to the "Timestamp" column
  Serial.write(27);
  Serial.print("left.");

  // Wait some ms
  delay(100);

}
```